Modeling the Performance of Large-Scale Systems

Darren J. Kerbyson, Adolfy Hoisie, Harvey J. Wasserman

Performance and Architectures Laboratory (PAL),
CCS-3, Los Alamos National Laboratory, New Mexico, 87545, USA
djk@lanl.gov

Abstract. Performance modeling can be used throughout the development, deployment, and maintenance of system hardware and application software. In this work we illustrate three uses of performance modeling on large-scale systems: the verification of performance during system installation, the comparison of two large-scale systems, and the prediction of performance on possible future architectures. We detail how a performance model gave an expectation of the performance of ASCI Q, a 20Tflop system recently installed at Los Alamos. A comparison between ASCI Q and the Earth Simulator is also detailed resulting in the sizing of an AlphaServer system that has the same performance as the Earth Simulator. Our modeling approach is application centric. A detailed model is developed for each application of interest based on a static analysis of the code but parameterized in terms of its dynamic behavior.

1 Introduction

Performance modeling is an important tool that can be used by a performance analysist to provide insight into the achievable performance of a system. It is only through knowledge of the workload the system is to be used for that a meaningful performance comparison can be made. It has been recognized that performance modeling can be used throughout the life-cycle of a system, or of an application, from first design through to maintenance [1]. This includes:

Design: performance modeling can be used to quantify the benefits between alternatives when architectural details are being defined.

Implementation: when a small system becomes available, perhaps a prototype, modeling can be used to provide an expected performance for larger scale systems.

Procurement: performance modeling can be used to compare competing systems. Measuring performance may not be possible on such systems due to either the scale of the system required being larger than anything available, or due to the system using next generation components which are not yet available.

Installation: performance modeling can provide an expected level of performance, and hence verify that a system is correctly installed and configured.

Upgrade or Maintenance: performance modeling can quantify the impact on possible upgrades prior to them being implemented.

Recent work at Los Alamos National Laboratory (LANL) has demonstrated the use of performance modeling in all of the above situations, for instance: in the early design of systems, during the procurement of ASCI purple (expected to be a 100-Tflop system to be installed in 2004/5), and to explore possible optimizations in applications prior to implementation [2]. In this work we show how performance modeling was used to verify the performance of ASCI Q during installation, to compare large-scale systems taking as example the Earth Simulator compared with ASCI Q, and to consider the performance of possible future systems. The details contained in this work build upon an analysis of the first stage of ASCI Q installation [3], and an initial performance comparison between the Earth Simulator and ASCI Q [4].

1.1 Performance Modeling

Our view is that a model should provide insight into the performance of the application on available as well as future systems. The development of performance models should also mirror the development of the application and/or system. As details are refined through implementations, the model should also be refined.

The approach we take is application centric. It involves the understanding the processing flow in an application, the key data structures, how they use and are mapped to the available resources, and also the effects of scaling. An analytical performance model of the application is constructed from this understanding. The aim is to keep the model of the application as general as possible but parameterized in terms of the application's key characteristics. The model is based on a static analysis of the code and is parameterized in terms of the code's dynamic behavior - i.e. those features which are not known through a static analysis.

Hardware characteristics should not be part of the application model but rather be specified as a separate component. Making a clear separation between software and hardware aspects is a key aspect in the modeling approach. Thus, once a model for particular system has been developed, it can be used to examine performance on a multitude of applications. Similarly, application performance can be compared across systems without any alteration to the application model. For instance a part of the system model will encapsulate the details of inter-processor communication and be parameterized in terms of the message size. The actual communication model may take the form of a simple linear analytical expression in terms of latency and bandwidth, or be more complex. Hardware characteristics may use measurements made by micro-benchmarks, or specified by other means.

This approach of hardware and software separation has been used in many modeling activities include the PACE system [1] for high performance computing, and INDY [5] for E-commerce based applications. An alternative modeling approach is to collect a trace of the application activity at run-time and to effectively replay it later within a modeling environment using characteristics of the system being modeled. These approaches tend to limit the performance space that can be explored as the processor count and problem characteristics are inherently contained in the trace. Such approaches include: Dimemas [6], the Trace Model Evaluator at LANL, and also the approach taken at San Diego Supercomputing Center [7].

The workload that has been used to analyze systems in this work includes: S_N deterministic particle transport on both structured meshes (Sweep3D) [8] and unstructured meshes [9], an adaptive mesh hydro code (SAGE) [2], and a non-deterministic Monte-Carlo particle simulation code (MCNP) [10]. A performance model for each of these applications has already been developed and all have been validated on a range of large-scale systems showing high prediction accuracy. This work focuses on the use of performance models of these applications. Details on the models are not included here as they have already been published.

An overview of the systems used in this work (ASCI Q and the Earth Simulator) is given in Section 2. The use of the SAGE performance model during the installation of ASCI Q is detailed in Section 3. A performance comparison between the Earth Simulator and ASCI Q is given in Section 4 which leads to the calculation of an equivalently sized Alpha system that has the same performance as the Earth Simulator. In section 5 the performance models of SAGE and Tycho are used to explore the performance that may be achieved on possible future systems.

2 ASCI Q and the Earth Simulator

ASCI Q and the Earth Simulator are currently the two largest systems in the top500 list of supercomputers. Their main characteristics are compared below.

The Earth Simulator: consists of 640 nodes inter-connected by a single stage 640x640 crossbar network [11]. Each node is an SMP composed of 8 arithmetic processors (AP), a shared memory of 16GB, a remote access unit (RCU) connecting the node to the crossbar, and an I/O processor (IOP). Each AP contains 8 vector units each with 8 pipes, and a 4-way super-scalar processor operating at 500MHz. Each is connected to 32 memory units with a bandwidth of 32GB/s (256GB/s aggregate within a node). The peak performance of an AP is 8-Gflops. The peak inter-node communication bandwidth is 16GB/s in each direction. The minimum latency for an MPI level communication is quoted as 5.2µsec within a node, and 8.6µsec between nodes [12]. It should also be noted that the Earth Simulator lead to the development of the NEC SX-6 product line [13]. An Earth Simulator node has better memory performance than that of the NEC SX-6, but a smaller capacity.

ASCI Q: is the latest ASCI system recently installed at LANL [14]. It consists of 2048 HP AlphaServer ES45 nodes. Each node is an SMP containing four 21264D EV68 Alpha microprocessors operating at 1.25-GHz with 64KB L1 instruction and data cache, 16MB unified L2 cache, and 16GB main memory. A peak memory bandwidth up to 8GB/s is possible within a node using two 256-bit memory buses running at 125-MHz. Four independent 64-bit PCI buses (running at 66MHz) provide I/O. Nodes are interconnected using two rails of the Quadrics QsNet fat-tree network. The peak bandwidth achievable on an MPI level communication is 300MB/s with a typical latency of 5µsec. The latency increases slightly with the physical distance between nodes. A detailed description of the Quadrics network can be found in [15].

Table 1. System Characteristics.

	Earth Simulator (NEC)	ASCI Q (HP ES45)
Year of Introduction	2002	2003
System Peak Performance (Tflops)	40	20
Node Architecture	Vector SMP	Microprocessor SMP
System Topology	single-stage crossbar	Quadrics fat-tree
Number of nodes	640	3072 (Total)
Processors - per node	8	4
- system total	5120	12288
Processor Speed	500 MHz	1.25 GHz
Peak speed - per processor	8 Gflops	2.5 Gflops
- per node	64 Gflops	10 Gflops
- system total	40 Tflops	30 Tflops
Memory - per node	16 GB	16 GB
- per processor	2 GB	4 GB
- system total	10.24 TB	48 TB
Memory Bandwidth (peak)		
- L1 Cache	N/A	20 GB/s
- L2 Cache	N/A	13 GB/s
- Main (per processor)	32 GB/s	2 GB/s
Inter-node MPI - Latency	8.6 µsec	5 µsec
- Bandwidth	11.8 GB/s	300 MB/s

The characteristics of the two systems are summarized in Table 1. It is clear that the peak performance of the Earth Simulator exceeds that of ASCI Q by a factor of two, and also that the main memory bandwidth is far higher. The inter-node communication performance in terms of latency is worse on the Earth Simulator but its bandwidth is a factor of 40 higher. Note that the inter-node MPI communication performance listed in Table 1 is based on measured unidirectional inter-node communication performance per network connection reported elsewhere.

3 Validating ASCI Q Performance during Installation

In this section we show how performance modeling was used during the installation of ASCI Q at LANL to verify that an acceptable performance level was being achieved. During installation two significant hardware upgrades were performed: the PCI bus of each node was upgraded to 66-MHz (from 33-MHz), and the processors were upgraded, from 1-GHz to 1.25-GHz with an increased 16MB L2 cache. The PCI bus speed determines the available bandwidth between the Quadrics NIC and processor memory and can have a significant impact on performance. Thus during the installation there were actually three configurations of processors and PCI bus speeds: initially a 1-GHz processor with a 33-MHz bus, followed by a 1-GHz processor with a 66-MHz bus, and finally a 1.25-GHz processor with a 66-MHz bus.

Table 2. Summary of Test Conditions.

Date	PCI bus (MHz)	PE Speed (GHz)	system (# nodes)	Performance issues
9-Sep-01	33	1	128	Some faulty nodes and communication links resulted in poor communication performance
24-Sep-01				Faulty hardware replaced, still poor communications
24-Oct-01				OS patch improved Quadrics Performance
04-Jan-02	66	1	512	PCI bus upgraded; SAGE performance at pre-Oct-24 (not all nodes successfully upgraded)
02-Feb-02				All nodes upgraded but some configured out lowering performance in collective communications
20-Apr-02				All nodes configured in, collectives improved
21-Sep-02	66	1.25	1024	1st segment of ASCI Q. Performance lower than expected on PE counts > 512
25-Nov-02				2nd segment of ASCI Q, consistent with 1st segment
27-Jan-03				Impact of operating system events reduced, performance significantly improved
1-May-03	66	1.25	2048	First test of full-system

The performance model of SAGE was used to provide expected performance for each of the three installation configurations using a problem size of 13,500 cells per processor. These predictions are shown in Figure 1. The following observations can be noted: 1) the SAGE predictions assume weak scaling and hence the time should ideally be constant across all processor configurations; 2) the model predicts that the two-fold improvement in PCI bus speed results in only a 20% performance improvement in the code overall; 3) the 25% improvement in processor clock speed results in a 22% performance improvement overall; 4) the cycle time is predicted to plateau above 512 processors – this is a characteristics of SAGE and occurs when all gather/scatter communications are out-of-node.

SAGE performance was measured throughout the installation process as summarized in Table 2. The three distinct configurations are clearly shown: initial hardware of 1-GHz processors with a 33-MHz PCI bus (tested between Sept and Oct '01), 1-GHz processors with a 66-MHz bus (tested between Jan and April '02), and the final hardware of 1.25-GHz processors with a 66-MHz bus (tested between Sept '02 and May '03). During the installation the system also increased in size.

The performance of SAGE was measured after the installation of the initial hardware: as soon as the machine was up and running (Sept. 9th), after an O/S upgrade and faulty hardware was replaced (Sept. 24th), and after an O/S patch (Oct. 24th). The upgrades that were most significant in terms of performance included bug fixes to the Quadrics resource scheduling software and O/S patches that affected the priority of a process that determined communication rail allocation. These three sets of measurements are compared with the model predictions in Figure 2.

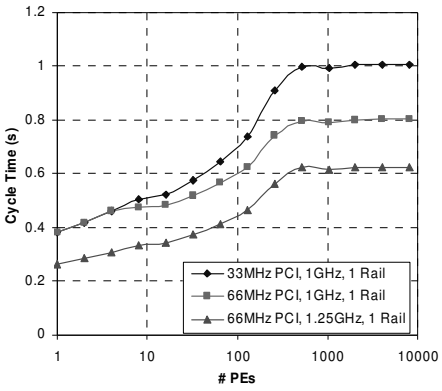

Fig 1. Performance prediction of SAGE on configurations of ASCI Q during installation

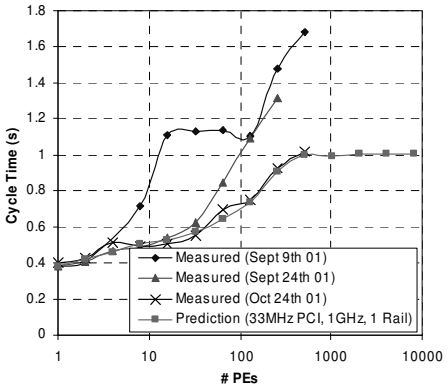

Fig 2. Measured performance of SAGE (33-MHz PCI bus, 1-GHz processors)

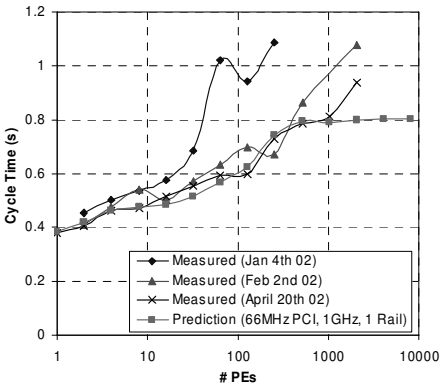

Fig 3. Measured performance of SAGE (66-MHz PCI bus, 1-GHz processors)

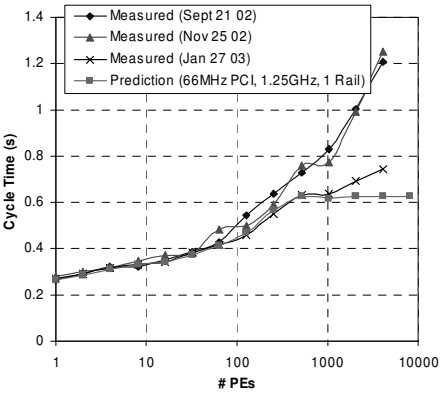

Fig 4. Measured performance of SAGE (66-MHz PCI bus, 1.25-GHz processors)

The corresponding measurements and model prediction based on the 1-GHz processors after the PCI bus was upgraded to 66-MHz are shown in Figure 3. Initially (Jan 4th), not all nodes ran at 66-MHz. By Feb 2nd this had been resolved; however, not all nodes were available for testing. The Quadrics QsNet requires contiguous nodes in order to use its hardware-based collective operations. By April 20th all nodes were available and SAGE achieved the performance predicted by the model for all configurations except for the largest count of 512 nodes (2048 PEs).

The performance of the system was again measured after upgrading the processors to 1.25-GHz, and the system increasing in size first to separate two segments of 1024 nodes each and then to a combined 2048 node system. The first measurements made on each of the two segments (Sept. 21st, and Nov. 25th) resulted in performance consistent with each other. However, a major performance issue was identified concerning the impact of the operating system. This resulted in very poor performance on applications with high synchronization requirements. These effects were mini-

mized by reducing the number of operating system daemons, reducing the frequency of some monitoring activities, and configuring out 2 nodes per 32-node cluster [16]. The performance measured after this optimization (Jan. 27th) resulted in much improved performance, close to the model predictions on the highest processor counts. The minimization of the operating system impact is ongoing.

The differences between the model predictions and the final set of measurements obtained for each of the three installation phases is small in all but the case of the largest processor counts. The performance on the largest configurations are expected to improve after further O/S optimizations. The average difference between the final measurements and model predictions across all the configurations is 3.7%.

Figures 2, 3 and 4 show that only after all the upgrades and system debugging had taken place that the measurements closely matched the expected performance. When the measured data matched the modeled data there was some confidence in the machine performing well. Without the model, it would have been difficult to know conclusively when to stop debugging, or more importantly when not to. When differences did occur between the model and measurements, microkernel benchmarks were run on the computational nodes and the communication network to help identify the source of the problem. This was especially important in minimizing the impact of the O/S that affected performance on very large processor counts [16].

4 Large-Scale System Comparison

In this section models of two codes representative applications of the ASCI workload are used to compare the performance of the Earth Simulator and ASCI Q: SAGE and Sweep3D. Three sets of analysis are included below:

1) the performance of SAGE and Sweep3D on ASCI Q and the Earth Simulator
2) a comparison of ASCI Q and Earth Simulator performance
3) sizing of an equivalent performing Alpha system, to the Earth Simulator, for an assumed combined workload consisting of 60% Sweep3D and 40% SAGE.

In these comparisons both SAGE and Sweep3D have been assumed to be used in a weak-scaling mode in which the sub-grid sizes on each processor remain a constant. However, since the main memory per processor is different between the two systems sub-grid sizes per processor were used which are in proportion to the main memory size of 4GB/processor on ASCI Q, and 2GB/processor on the Earth Simulator. This corresponds to the applications using all the available memory. Both the weak scaling scenario and the use of as much memory as possible are typical of they way in which large-scale ASCI computations are performed. The number of cells per processor assumed for SAGE were 37,500 on ASCI Q, and 17,500 on the Earth Simulator. The sub-grid sizes assumed for Sweep3D were 12x12x280 (40320 cells) on ASCI Q and 8x8x280 (17920 cells) on the Earth Simulator.

Both performance models are a function of the time to compute a sub-grid of data on a single processor. This time has been measured on ASCI Q but is unknown on

the Earth Simulator. To circumvent this, in this analysis we predict the runtime of the applications on the Earth Simulator for a family of curves. Each curve assumes a speed of either: 5%, 10%, 15%, 20%, 25%, or 30% of single processor-peak. This also has the advantage in that the analysis will remain valid over time: since codes can be optimized for a particular system, the percentage of single processor-peak may improve over time.

The performance of both SAGE and Sweep3D has been measured on ASCI Q. SAGE currently achieves approximately 10% of single processor-peak performance on the AlphaServer ES45 used in ASCI Q, and Sweep3D achieves approximately 14%. Both codes may need to be modified (at worst re-coded) in order to take advantage of the vector processors in the Earth Simulator. However, none of these codes is particularly tuned for the architectural features of the RISC architectures either, particularly the on-chip parallelism and the memory hierarchy [17]. Low levels of single processor-peak performance have currently been observed on the NEC SX-6 for both SAGE and Sweep3D. These are discussed further below.

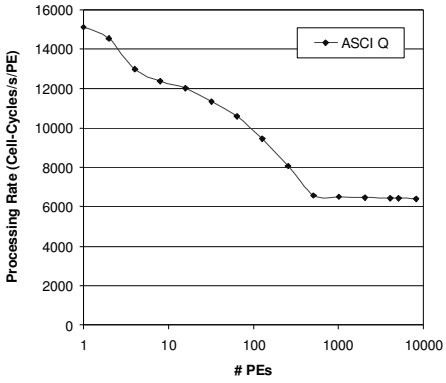

Fig 5. SAGE performance (ASCI Q)

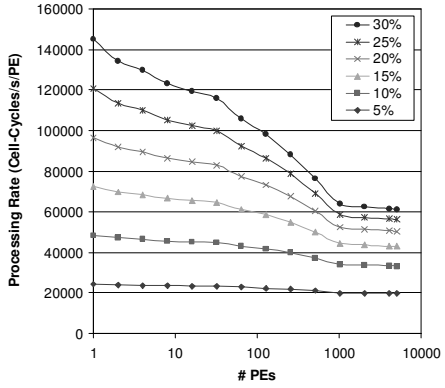

Fig 6. SAGE performance (Earth Simulator)

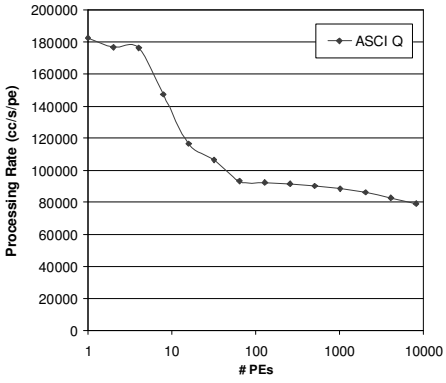

Fig 7. Sweep3D performance (ASCI Q)

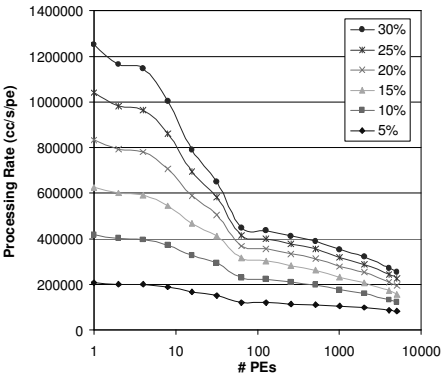

Fig 8 Sweep3D performance (Earth Simulator)

4.1. Performance of SAGE and Sweep3D on ASCI Q and the Earth Simulator

The performance of SAGE on ASCI Q is shown in Figure 5, and for the Earth Simulator in Figure 6. The metric used in this analysis is the cell processing rate – the number of cells processed per second per processor (cc/s/pe). The cycle time as used in Section 4 is simply the cells per processor divided by cc/s/pe. It can be seen that the value of cc/s/pe decreases with increasing processor count due to the increase in parallelism costs. A similar comparison is given in Figures 7 and 8 for Sweep3D.

4.2 Earth Simulator and ASCI Q Relative Performance.

The performance of the Earth Simulator is compared with that of ASCI Q for SAGE in Figure 9. This shows the relative processing rate between the two systems on an equal processor count basis. Recall that one Earth Simulator processor has an 8Gflops peak performance, and each ASCI Q Alpha processor has a 2.5Gflops peak performance. The relative advantage of the Earth Simulator based on peak performance alone is a factor of 3.2 (indicated by a single horizontal line in Figure 9). A value greater than 3.2 indicates a performance advantage of the Earth Simulator compared to ASCI Q over and above the ratio of single processor-peak performance. Figure 10 shows a similar analysis for Sweep3D.

It can be seen that the relative performance between the two systems is very dependent on the application. For instance, the relative performance of SAGE on the Earth Simulator is higher than the processor-peak ratio in nearly all cases (between a factor of 3 and 9). However, the performance of Sweep3D is less than the process-peak ratio on the higher processor counts (between a factor of 1 and 3). The poorer performance of the Earth Simulator on Sweep3D is due in part to the communication requirements in this application being largely latency-bound, and also in part due to the difference in the scaling behavior of the different problem sizes as a result of the difference in memory capacities.

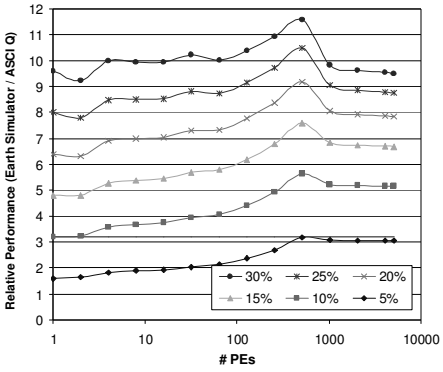

Fig 9. Relative Performance (SAGE)

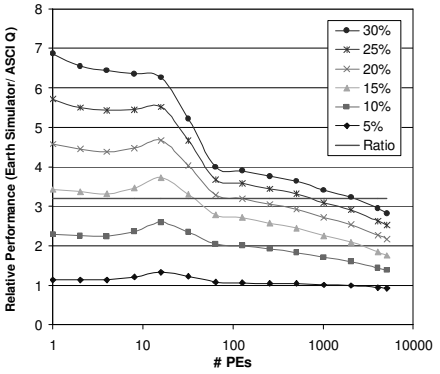

Fig 10. Relative performance (Sweep3D)

4.3 Sizing an Equivalent Performing Alpha System to the Earth Simulator

Using the performance models the size of an Alpha system that would achieve the same performance as the Earth Simulator can be determined. In this analysis, a hypothetical workload of 60% Sweep3D, and 40% SAGE is assumed. The size of the Alpha system is calculated in terms of its peak-Tflop rating. The result of this calculation is shown in Table 3. The same range of single-processor peak percentages for the Earth Simulator is used as before. Recall that an AlphaServer ES45 node has a peak performance of 10GFlops and thus the number of ES45 nodes is simply the values in Table 5 multiplied by 100.

Table 3. Peak-Tflop rated Alpha system required to achieve the same performance as the Earth Simulator using an assumed application weighting of 40% SAGE and 60% Sweep3D.

		SAGE % of single-processor peak					
Sweep3D		5%	10%	15%	20%	25%	30%
	5%	23	34	42	48	53	57
	10%	27	38	47	53	57	61
	15%	30	41	50	56	60	64
	20%	34	45	53	59	64	68
	25%	38	49	57	63	68	72
	30%	41	52	60	66	71	75

Performance of SAGE has been measured on a single NEC SX-6 node achieving 5% of single-processor peak. Sweep3D achieved a very low percentage. Thus at the present time the equivalent sized Alpha system would have a peak of approximately 23Tflops and ASCI Q, at 20Tflops, achieves a comparable performance on the two representative ASCI applications. The information in Table 3 could be combined with the cost to determine which system has the better price-performance.

5 Exploring the Performance of Possible Future Systems

Performance models can be used to explore the possible performance that may be observed on future systems. In the following analysis we assume a system architecture similar to that of the current Alpha-server ES45 in that a number of nodes are interconnected in a fat-tree topology. The nodes are also assumed to contain four processors. The performance of two applications, SAGE and Tycho, are examined on this hypothesized system assuming that each of the: computational performance, network bandwidth, and latency have improved in performance by a factor of eight. This is done by altering the system input parameters to the performance models.

Figure 11 shows the performance of SAGE in terms of its cycle-time when the performance of individual system components is improved. The problem size used in this analysis was 35,000 cells per processor. The overall performance improvement from these sub-system improvements is shown in Figure 12 along with that that could be obtained by just doubling the processor count in the system.

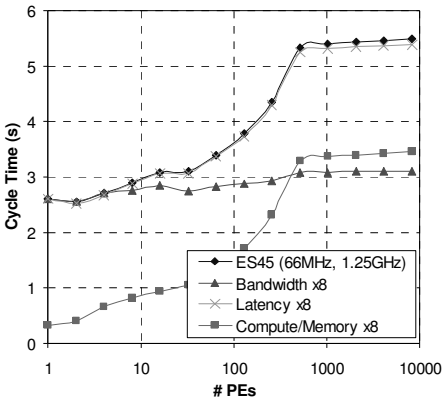

Fig 11. Performance prediction of sub-system improvements (SAGE)

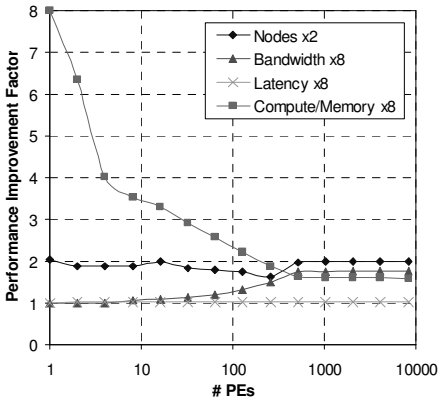

Fig 12. Performance improvement factor of sub-system improvements (SAGE)

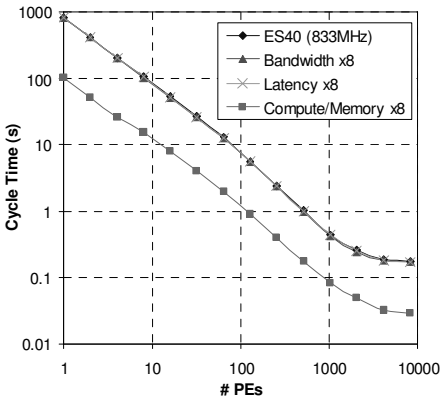

Fig 13. Performance prediction of sub-system improvements (Tycho)

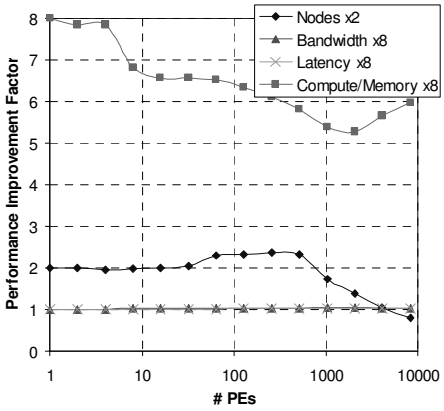

Fig 14. Performance improvement factor of sub-system improvements (Tycho)

It can be seen that SAGE is computation/memory bound on small processor counts (<256), and thus improving just the computational performance of a node gives the greatest overall performance improvement in this region. On larger processor counts the greatest improvement would be gained by simply doubling the number of processors. This is due to the plateau in the SAGE cycle time occurring on larger processor counts.

A similar analysis is depicted in Figures 13 and 14 for Tycho. This is performed for an unstructured mesh with 1,000,000 cells. The form of the curves in Figure 13 is different to that of SAGE in Figure 11 due to a strong scaling characteristic of Tycho. Tycho remains compute bound throughout. As can be seen in Figure 14, the best performance improvement would be gained by an increase in the computational performance of the nodes. These curves do not directly indicate the efficiency of the

calculation – on larger processor counts the efficiency of Tycho decreases. It can be seen that using more than 4000 processors is not beneficial as the cycle time starts to increase. This is due to a characteristic of the scaling behavior of Tycho.

Studying the expected performance on possible future systems illustrate one importance use of performance modeling – that is to explore performance prior to implementation. It should be noted however, that the performance improvements vary from application to application. If all elements of the workload to which a system is to be used for are known in advance, the information from many performance models can be combined to give quantitative information on the expected performance of the full workload. Such information is useful in system architecture design.

6 Conclusions

Our team's research over the last few years has focused on the development of analytical performance models of the ASCI workload. ASCI has a critical need for information on how best to map a given application to a given architecture, and performance modeling is the only means by which such information can be obtained quantitatively. Our approach has been successfully demonstrated for several applications including S_N deterministic transport [8,9], Monte-Carlo particle simulation [10], and an adaptive mesh hydro code [2].

In this work we have demonstrated the use performance modeling in several important areas. Performance models were used in advance of the installation of ASCI Q to provide an expected level of performance that the system should achieve. During installation, deviations from the expected performance were identified and used to determine if further refinements in the system configuration or debugging was necessary. Implementation milestones tests related to ASCI Q contractual obligations were based partially on the comparison of observed data with predictions from the performance model, in a manner similar to the process described in Section 3.

Comparing measured performance of large-scale systems across many performance scenarios is often not possible either due to limitations in system access or, in the case of procurement, due to no suitable system being available for testing. We have illustrated how performance models can be used to compare large-scale system performance, using ASCI Q and the Earth Simulator as example. The systems were compared using a representative workload. This lead to the sizing of an AlphaServer system that has the same achievable performance as the Earth Simulator. Through this analysis quantitative information has been added to the long debated discussion over the use of COTS based components, which have lower performance but are often cheaper, in comparison to the use of more expensive specialized processors.

We expect that ASCI platforms and software will be performance engineered, and that models will provide a means for this. The models can play a role throughout a system's lifecycle: starting at design when no hardware is available for measurement, in procurement for the comparison of systems, through to implementation / installation, and to examine the effects of updating a system over time. As each point the performance model provides an expectation of the achievable performance.

Acknowledgements

Los Alamos National Laboratory is operated by the University of California for the National Nuclear Security Administration of the US Department of Energy.

References

1. Nudd, G.R., Kerbyson, D.J., Papaefstathiou, E., Perry, S.C., Harper J.S., Wilcox, D.V.: PACE: A Toolset for the Performance Prediction of Parallel and Distributed Systems. Int. J. of High Performance Computing Applications, **14** (2000) 228-251
2. Kerbyson, D.J., Alme, H.J., Hoisie, A., Petrini, F., Wasserman, H.J., Gittings, M.: Predictive Performance and Scalability Modeling of a Large-Scale Application. In Proc. SuperComputing, Denver (2001)
3. Kerbyson, D.J., Hoisie, A., Wasserman, H.J.: Use of Predictive Performance Modeling During Large-Scale System Installation. To appear in Parallel Processing Letters (2003).
4. Kerbyson, D.J., Hoisie, A., Wasserman, H.J.: A Comparison Between the Earth Simulator and AlphaServer Systems using Predictive Application Performance Models. In Proc. Int. Parallel and Distributed Processing Symposium (IPDPS), Nice France (2003)
5. Papaefstathiou, E.: Design of Performance Technology Infrastructure to Support the Construction of Responsive Software. In Proc. 2ns ACM Int. Workshop on Software and Performance (WASP), Ottawa Canada, (2000) 96-104
6. Girona, S., Lambarta, J., Badia, R.M.: Validation of Dimemas communication model for MPI collective operations. In Proc. EuroPVM/MPI, Balatonfured, Hungary (2000)
7. Carrington, L., Snavely, A., Wolter, N., Gao, X.: A Performance Prediction Framework for Scientific Applications. Proc. Int. Conference on Computational Sciences (ICCS), LNCS, Springer-Verlag (2003)
8. Hoisie, A., Lubeck, O., Wasserman, H.: Performance and Scalability Analysis of Teraflop-Scale Parallel Architectures using Multidimensional Wavefront Applications. Int. J. of High Performance Computing Applications, **14** (2000) 330-346.
9. Kerbyson, D.J., Pautz, S.D., Hoisie, A.: Performance Modeling of Deterministic Transport Computations. In Performance Analysis and Grid Computing, Kluwer (2003).
10. Mathis, M., Kerbyson, D.J.: Performance Modeling of MCNP on Large-Scale Systems. In Proc. Int. Conference on Computational Sciences (ICCS), LNCS, Springer-Verlag (2003)
11. Watanabe, T.: A New Era in HPC: Single Chip Vector Processing and Beyond. In proc. NEC Users Group meeting, XIV (2002)
12. Kitawaki, Yokokawa, M.: Earth Simulator Running. In Proc. Int. Supercomputing Conference (ICS), Heidelberg, June (2002)
13. The NEC SX-6, product description, NEC Corporation, http://www.sw.nec.co.jp/hpc/sx-e
14. ASCI Q, http://www.lanl.gov/asci
15. Petrini, F., Feng, W.C., Hoisie, A., Coll, S., Frachtenberg, E.: The Quadrics Network: High-Performance Clustering Technology. IEEE Micro 22 (2002) 46-57
16. Petrini, F., Kerbyson, D.J., Pakin, S.: The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q, in Proc. of SuperComputing, Phoenix (2003)
17. Goedecker, S., Hoisie, A.: Performance Optimization of Numerically Intensive Codes, SIAM Press (2001)